# Heart Machine Control Circuit

The Heart Machine is a sculpture that provides a sensory, interactive experience for one to 25 participants. The underlying premise of the sculpture is to explore the interactions, both human/machine and human/human. In order to achieve a non trivial reaction from the machine, multiple participants, working together are required. It is this coming together of friends and strangers solving simple puzzles that both surprises and delights the participants. Cooperation is rewarded.  
The Heart Machine consists of four arteries, with five inputs per artery. At its simplest, when all five of the inputs are enabled, the Flame Effect at the end of the Artery is ignited.

Although the overall interaction is still to be designed, consider a few scenarios:

1. The machine seeks human input in order to function. All the nodes within the artery must be ‘pressed’ in order to complete the ‘circuit’. To entice people to complete the circuit, the Heart Machine will pulse the inputs that remain un-pressed.
2. A variation, suggested by Jason, is to implement a ‘light chaser’ in which the input nodes are pulsed in sequence, moving from the Heart Machine in the center out towards the artery. If a participant is able to press the input when the light is present, they can trap it, enabling a participant at the next station to trap the next sequence, and so on until all inputs are enabled.
3. Through the use of multiple inputs coupled with different colored lighting effects, it is possible to create more advanced sequences. These multiple inputs might be used to create more sophisticated interactions when a smaller number of participants are present.

A number of parameters can be adjusted to control the flame effects. The duration of the burn can be adjusted. In addition, the flame can be pulsed, yielding both control over the height of the flame as well as changing the Flame Effect’s ‘tone’. There is currently an active discussion around how to adjust flame color and flame tone that may also require additional control.

## Guiding Principles

The following Guiding Principles are to be used as a framework for decision making:

* Multiple participants are required to achieve a given effect.
* The Heart Machine experience is additive. Directed participation is rewarded.
* The Heart Machine shall evolve over time. New experiences need to be supported easily.

## Basic Design

The Heart Machine consists of four ‘circuits’. Each circuit is comprised of five input node and one output flame effect node. The fifth input node in each circuit is coincident with the output flame effect node.

Each input node in the Heart Machine has a set of input switches with some form of visual feedback. Each input node also has two variable intensity lights (one for the palm and one for the structure). Input nodes can submit messages when an input event occurs.

Each flame effect output node has two outputs to control burners as well as inputs to measure temperature and pressure. In addition, two variable intensity lighting channels are provided.

The nodes are situated on a bus. Each node is uniquely addressable. The overall experience is controlled by a central controller which also acts as the bus master. The central controller changes system state by reading from, and writing values to, nodes on the bus. Input nodes can originate messages, but these messages are lower priority than messages that the controller sends out.

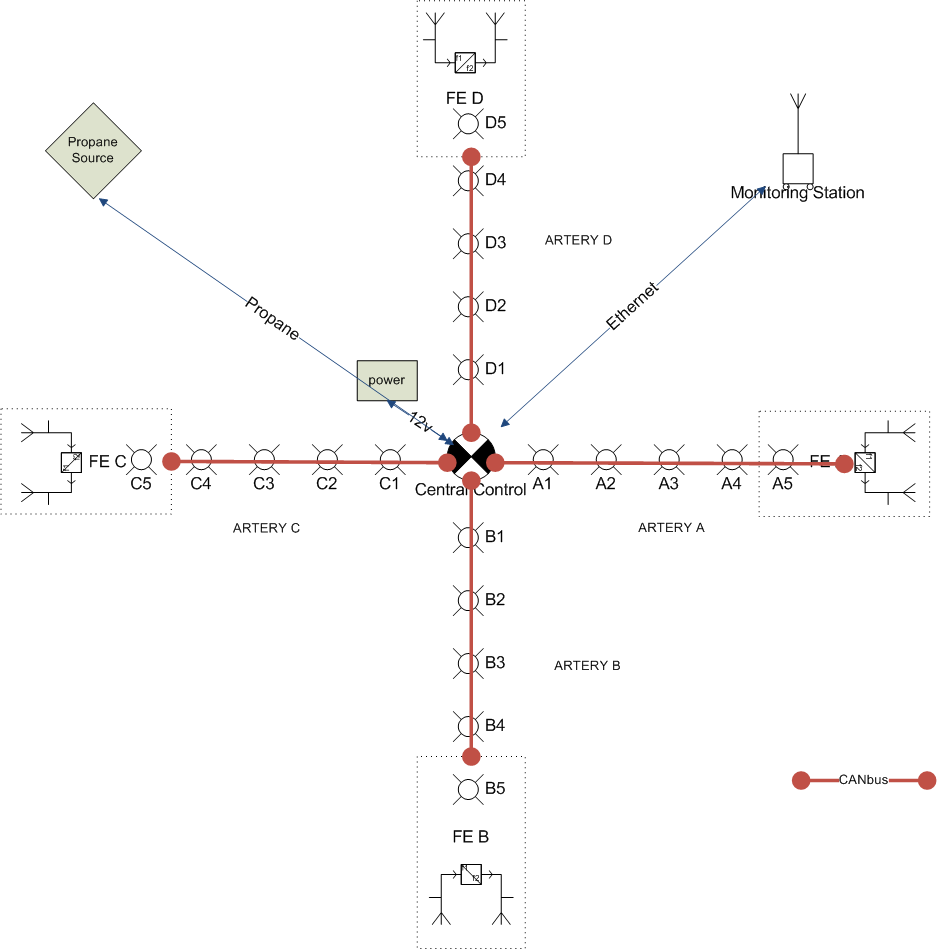


Figure - Top down schematic / physical layout of the HM. The sculpture is 50-60’ across

## Constraints

* Very low cost per node
* Very reliable messaging required
* Simple wiring (bus preferred over separate cable runs)
* Standards complaint if possible
* Easy to program experiences; easy to change these experiences on the fly.
* KISS. Very simple electronics, with as little code on each node as possible

## CANBus

The CANbus protocol has been selected to network the nodes. Controller–area network (CAN or CAN-bus) is a bus standard designed to enable microcontrollers and devices to communicate with each other without a host computer. CAN is a message based protocol, designed specifically for automotive applications but now also used in other areas such as industrial automation and medical equipment. It is designed for high reliability in hostile environments. Most importantly for our control applications, it features guaranteed message delivery.

Due to its widespread use in industrial applications, low-cost solutions CAN-bus are available. We propose to use the Microchip MCP2515 CAN bus node with a MCP25050 I/O expansion chip. This results in a minimal three chip circuit that can be readily constructed using thru-hole circuit board techniques to keep costs down. The MCP25050 device features a number of peripherals, including 8-bits of digital I/Os, four-channel s of 10-bit A/D, with input threshold triggers, and 2 PWM outputs.

It is proposed that we also include an AVR ATMega328 CPU on the board. This chip is the heart of the popular Arduino controllers and can be configured to communicate over the CAN base using SPI should we choose. The ATMega will be used to sense the capacitive sensors and provide additional lighting control. Arduino libraries for the MCP2515 CAN controller are readily available.

Parts cost is ~$15 not including circuit board and connectors.

## General Node Design

Each node will consist of the three primary CANbus chips (transceiver, controller, I/O expander) and the AVR CPU. Additional power for driving high wattage LEDs will be implemented using a ULN2003 or equivalent if required. Two MOSFETs will be present on the board to drive the solenoids and the high-current LEDs.

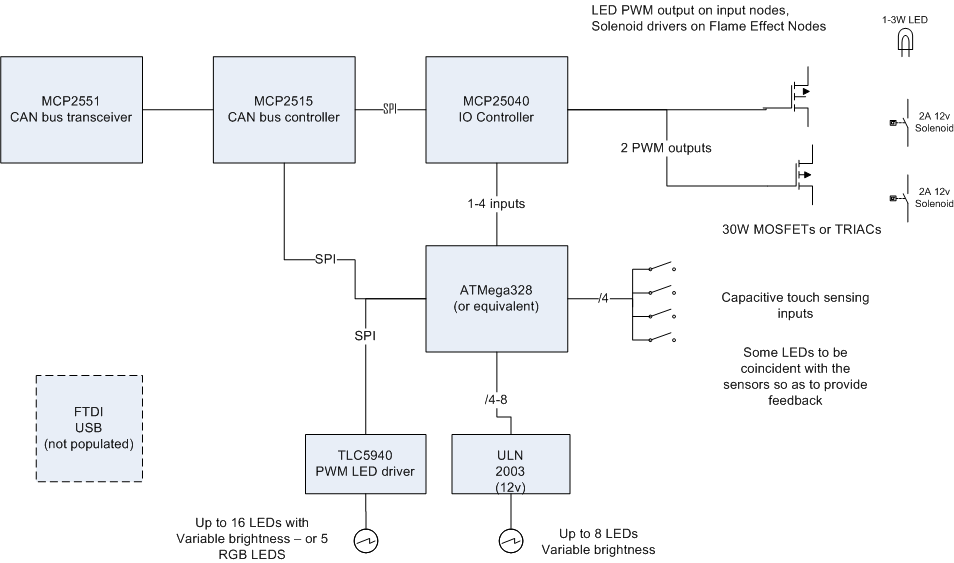


Figure - basic block diagram of a Heart Machine node. The 5940/ULN2003 and FTDI are optional

In addition, board space will be left for a TI TLC5940 (16 channels of PWM with current limiters for LED drivers).

### Quick CHIP CAPABILITY Summary

#### MCP2515

Talk CANbus protocol natively. Has an ID programmed into its EPROM.

presents an SPI connection to both the MCP25050 and the ATMega328

#### MCP25050

Base I/O device as it has built-in CAN control. Can be removed later if desired and replaced with The AVR. Configured through SPI by the MCP2515.

Shares 8-bits of GPIO with the following capabilities:

* two PWM outputs (up to 2 outputs, 10-bit resolution)
* A/D (up to 4 inputs, 10-bit)
* Programmable clock out
* Reset

All GPIOs can be configured to send a message on an edge trigger.

* If A/D used, then requires at least one of the GPIO for reference. The A/D can be configured to trigger on a specific threshold voltage.
* Each CAN IO expander appears as collection of registers on the network.
* All CAN nodes receive all messages. Filters are used to determine which messages each node acts upon.
* Nodes can generate their own messages.
* One node is determined to be the bus master.

#### AVR ATMega328

Capacitive Sensing, LED Control, Future Expansion

* SPI interface (optionally interface this with the MCP2515)
* 32kb Flash
* 2kb RAM
* 16MHz 8-bit processor
* 14 I/O (6 PWM)
* 6 ADC
* Can be programmed in circuit

#### TLC5940

LED control (optional)

* Up to 16 channels of PWM output. Able to sink 100ma per output
* 28 pin DIP
* Controlled over SPI or through bitbanging

### Power

An onboard regulator will convert 12v to 3.3v. The semiconductors will consume < 100ma at 3.3v. LEDs and the MOSFETS will consume up to several watts and run off of the 12v.

### Cabling

CAT5 cabling will be used to provide communications. A separate 12v power bus using 14 gauge wire will run along each artery and power each device.

## Input Node

The Input Node is designed to attract attention and to sense input. Capacitive sensing is used so that proximity can be measured without touching. We believe we can adjust the capacitive sensing such we can adjust the brightness of LEDs based on proximity.

To increase the visual interest of the input board, we can backlight the individual sensors with a different colored LED. We can also take any input and use it to drive the UV side lighting. In this way, as you move your hand over the control surface, the surface can change color while still glowing.

Basic features:

* 2-4 digital inputs from the capacitive sensors
* Two high wattage attractors LED from PWM/MOSFET circuit
* 8 digital outputs driven by ULN2003 driven by PWM output from ATMega. We could expand this to 4 x RGB LEDs from TI chip if funds allow.

We should probably commission a separate project to create the lighting effects. A simple PIC microcontroller circuit such as this would be a good start: <http://picprojects.org.uk/projects/rgb/#Schematic_and_PCB_artwork>

### Capacitive Sensing

Use ATmega328 with QTouch for sensor input. The advantage of this approach is that we can have more PWM channels as well as a more general purpose board. We can borrow the Arduino circuit layout and simply extend the board with room for the sensor pads and LEDs.

<http://www.youtube.com/watch?v=m91dwBjfAng>

<http://vimeo.com/7771360>

## Output Node

The output nodes sit within the flame effect units. They need two output channels to drive the solenoids.

As we don’t have any input here, 6 bits of output are available through the ULN2003 to drive other lighting / controls.

Depending on the output configuration, we can use up to 4 channels of 12-bit ADC to measure temperature and/or pressure.

## Control Node

The controller is a CAN<->USB controller. Internally, this adapter uses the FTDI Serial->USB->FIFO interface. The driver appears to be a modified Serial interface to Windows. A modified DLL is provided to make communications efficient. The CAN bus is restricted to 1Mbit. Given that CANbus messages are 8 bytes, doing simple math, we can pump 1Mbit/10/8 = 12,500 messages. We have ~32 nodes in our system so in total we can exchange 400 messages/second. Clearly this is more than enough.

A key component of interactivity is response time. In certain scenarios, such as the tapping of a beat, we’ll want to be able to receive and process a set of inputs and send an output to the flame effect within < 1/10 of a second or 100 ms.

The input nodes will broadcast their inputs every time a touch is felt. These messages have a lower priority than system commands and output messages. Nodes buffer and continue to send their messages when the bus is free. We will have to make sure that holding down one’s hand on the input sensor does not result in a continuous stream of messages else we run the risk of flooding the network with superfluous messages.

The central controller will be hosted on a ruggedized (no fan, no moving parts) PC running a custom Java control program.

## Control Software

We need to be able to reconfigure the HM should an input node fail. Thus, at the lowest level, the controller software is responsible for mapping node inputs to nodes in the system graph. This is to enable the replacement/swapping out of nodes for maintenance/repair.

Inputs are sampled over time and the most recent X samples are maintained in memory. This will enable us to detect input patterns such as a large number of participants drumming, or a small number of users following a pattern set out by the HM.

### Input Sampling

Very preliminary…

Each input consists of an array which models inputs over time.

Although implemented as an interrupt driven system, it is likely easiest to visualize the process sequentially:

1. Inputs sampled
2. Attractor process runs on the inputs to try to determine where additional input is required. This involves looking at the historical inputs (last 1-5 seconds). A threshold is used to determine how precise we want to make the inputs. It may be sufficient that we’ve had an input in the last ½ a second to trigger the FE. This is especially important when we consider beat-input.
3. Send out FE updates
4. Send out LED updates
5. Age the inputs
6. Goto 1

The FE should be modeled separately. Depending on the ‘mode’ of the system, the FE may be more active. Parameters that we can play with include: frequency and amplitude of random ‘twitches’.

### Driving the FEs

The FE solenoids are driven by (to be determined) either direct digital I/O or pulse width modulator outputs. This provides us with control over two variables: the duty cycle (on to off) and the frequency. Given that we’re driving an electro-mechanical system that controls a valve, we are restricted to a certain range. Too fast or too slow a duty cycle will result in the flame being effectively on or off all of the time. Too fast a frequency will result in the solenoid being driven all of the time, albeit perhaps at a voltage that is below its switching threshold.

Hence, we will need to constrain the values that we send the FE controller board’s PWM driven outputs.

Per FE Parameters we can model

* Twitch (random firing)
* Heartbeat Amplitude
* Heartbeat Frequency
* Relationship between FE1 and FE2

System Wide FE Parameters we can model

* Mode – single FE/multi FE (call, response), swirl rate, swirl direction, swirl ramp function (set to specific level or use a function to determine amplitude)
* Model inner ring separately from outer ring?

## Monitoring Program

Written in Flash or Java

* Provides a schematic of the HM
* Each node displays its input and output characteristics
* Each node can have its inputs ‘virtually pressed’
* Shows FE parameters (temperature, pressure) if available
* Shows all LED output
* May also provide input/control over system (reset, etc.)
* May also be able to load new ‘behaviours’